Lógica de Programação

Módulo 01

Introdução | Algoritmos

Introdução

O que é lógica de programação? Navegando pela Internet encontramos diversas definições para o termo:

Lógica de programação é a organização coesa de uma sequência de instruções voltadas à resolução de um problema, ou à criação de um software ou aplicação.

Lógica de Programação é o modo como se escrevem programas de computador através de uma sequência de passos para executar uma ou várias funções, esta sequência é o algoritmo.

Lógica de programação é a técnica utilizada para desenvolver instruções em uma sequência para atingir determinado objetivo. É a organização e planejamento de instruções, em um algoritmo, com o objetivo de tornar viável a implementação de um programa ou software.

A lógica de programação (ou, por extensão, lógica computacional) é uma forma de organizar pensamentos que permite a tradução do raciocínio lógico humano para a linguagem das máquinas, permitindo que elas realizem alguma determinada tarefa.

Para aprender a desenvolver sistemas é necessário primeiro aprender Lógica de Programação. A Lógica de Programação é o alicerce sólido que você deve construir para ter uma boa base para desenvolver seus sistemas em qualquer que seja a linguagem de programação. Sem uma boa base em Lógica de Programação, você está fadado a não compreender totalmente o que está fazendo e não estará apto para resolver problemas de média ou alta complexidade, muitas vezes apresentando soluções não satisfatórias para os mesmos.

Comecemos então nossa jornada!

Linguagens de Programação

O que é uma linguagem de programação? Vejamos a definição da Wikipedia:

A linguagem de programação é um método padronizado, formado por um conjunto de regras sintáticas e semânticas, de implementação de um código fonte - que pode ser compilado e transformado em um programa de computador, ou usado como script interpretado - que informará instruções de processamento ao computador. Permite que um programador especifique precisamente quais os dados que o computador irá atuar, como estes dados serão armazenados ou transmitidos e, quais ações devem ser tomadas de acordo com as circunstâncias. Linguagens de programação podem ser usadas para expressar algoritmos com precisão.

O conjunto de palavras (lexemas classificados em tokens), compostos de acordo com essas regras, constituem o código fonte de um software. Esse código fonte é depois traduzido para código de máquina, que é executado pelo microprocessador.

Nós podemos classificar as linguagens de programação de várias formas diferentes. Vejamos algumas delas.

Classificação ACM

Existe uma associação chamada *Association for Computing Machinery* (Associação para Maquinaria da Computação) que foi criada em 1947. Ela foi a primeira sociedade científica e educacional dedicada à computação. Ela possui cerca de 80.000 membros e a sua sede está situada na cidade de Nova Iorque. A ACM mantém um sistema de classificação com os seguintes itens:

- Linguagens aplicativas, ou de aplicação
- Linguagens concorrentes, distribuídas e paralelas
- Linguagens de fluxo de dados
- Linguagens de projeto
- Linguagens extensíveis
- Linguagens de montagem e de macro
- Linguagens de microprogramação
- Linguagens não determinísticas
- Linguagens não procedurais
- Linguagens orientadas a objeto

- Linguagens de aplicação especializada
- Linguagens de altíssimo nível

Paradigma

Paradigma significa *um exemplo que serve como modelo; um padrão*. Quais são os paradigmas relacionados às linguagens de programação? Os paradigmas se dividem em dois grandes grupos: imperativo e declarativo.

Paradigma Imperativo

Os paradigmas imperativos são aqueles que facilitam a computação por meio de mudanças de estado. Se dividem em:

- Procedural ou Procedimental. Neste paradigma, os programas são executados através de chamadas sucessivas a procedimentos separados. Exemplos de linguagens deste paradigma são o Fortran, o BASIC e o Clipper.
- Estruturas de Blocos. A característica marcante deste paradigma são os escopos aninhados. Exemplos de linguagens deste paradigma são o Algol 60, Pascal e C.
- Orientado a Objetos. Este paradigma descreve linguagens que suportam a interação entre objetos. Exemplos de linguagens deste paradigma são C++, Java, Python, Delphi e Ruby.
- Computação Distribuída. Este paradigma suporta que mais de uma rotina possa executar independentemente. Um exemplo de linguagem deste paradigma é a linguagem Ada.

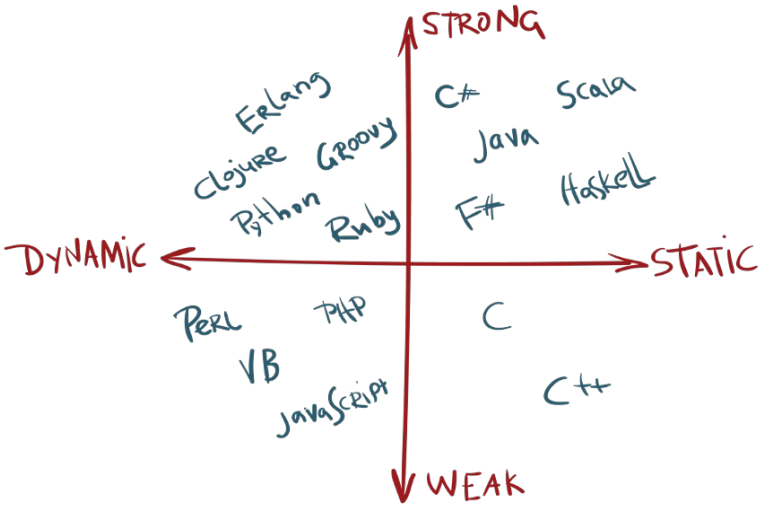
Paradigma Declarativo

Os paradigmas declarativos são aqueles nos quais um programa especifica uma relação ou função. Se dividem em:

- Funcional. Linguagens deste paradigma não incluem qualquer provisão para atribuição ou dados mutáveis. Na programação funcional, o mapeamento entre os valores de entrada e saída são alcançados mais diretamente. Um programa é uma função (ou grupo de funções), tipicamente constituída de outras funções mais simples. Exemplos de linguagens deste paradigma são Lisp, Scheme e Haskell.
- Programação Lógica. Este paradigma se baseia na noção de que um programa implementa uma relação ao invés de um mapeamento. Exemplos de linguagens deste paradigma são Prolog e Gödel.

Estrutura de Tipos

As linguagens de programação podem ser definidas de duas formas quanto a sua estrutura de tipos.



Fortemente ou Fracamente Tipada

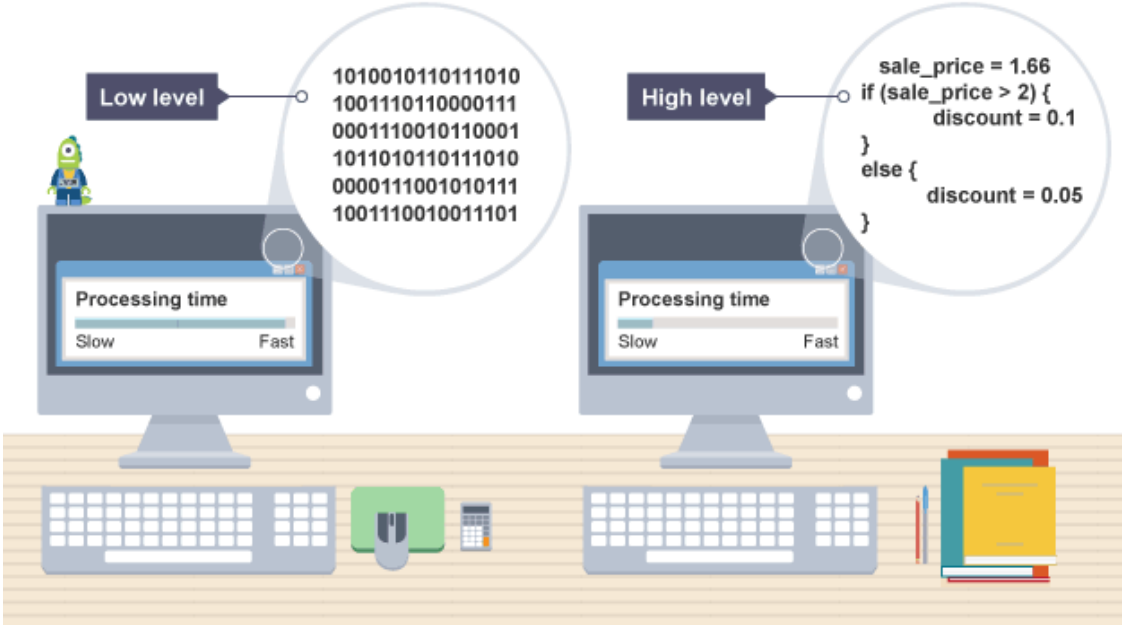
- Fracamente tipada, como PHP e Smalltalk, onde o tipo da variável muda dinamicamente conforme a situação.
- Fortemente tipada, como Java, Ruby e Python onde o tipo da variável, uma vez atribuído, se mantém o mesmo até ser descartada da memória.

Dinamicamente ou Estaticamente Tipada

- Dinamicamente tipada, como SNOBOL, APL, Awk, Perl, Python e Ruby, onde o tipo da variável é definido em tempo de execução.
- Estaticamente tipada, como Java e C, onde o tipo da variável é definido em tempo de compilação.

Abstração

Quando mencionamos abstração em relação às linguagens de programação, estamos nos referindo à proximidade da linguagem de programação para a linguagem humana e para a linguagem de máquina? Como assim? Quanto mais próximo da linguagem de máquina, menor é o nível de abstração e quanto mais próximo da linguagem humana, maior é o nível de abstração, de tal maneira que nos referimos a essas linguagens como: linguagens de baixo nível, linguagens de médio nível e linguagens de alto nível.



Normalmente utilizamos linguagens de alto nível para desenvolver nossos sistemas: Pascal, Java, C++, C#, Dart, PHP, Python etc.

Qual Usar?

Qual linguagem de programação você deve usar para aprender Lógica de Programação? Você pode receber muitas respostas diferentes para essa pergunta. Alguns fervorosos defensores de suas linguagens preferidas vão incentivar o uso da tal linguagem. Nossa sugestão é que você não use nenhuma linguagem de programação para aprender a lógica e é esse conceito que utilizaremos durante esse treinamento.

Estudaremos lógica de programação de tal maneira que, após o treinamento, você será capaz de aplicar os conceitos nas diversas linguagens de *paradigma imperativo.*

Algoritmos

Quando se estuda Lógica de Programação, um dos primeiros termos que aparecem é o "Algoritmo". Mas afinal, o que é um algoritmo? Vamos analisar novamente vários conceitos encontrados na Internet.

Um algoritmo é formalmente uma sequência finita de passos que levam à execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma sequência de instruções que dão cabo de uma meta específica. Estas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

É uma linguagem intermediária entre a linguagem humana e as linguagens de programação.

É utilizado para representar a solução de um problema.

Descrevem instruções a serem executadas pelos computadores.

É a especificação de uma sequência ordenada de instruções, finitas e não-ambíguas, que deve ser seguida para a solução de um determinado problema, garantindo a sua repetibilidade.

Você sabia que você utiliza algoritmos frequentemente no seu dia a dia? Nós aplicamos o conceito de algoritmo diariamente sempre que estabelecemos um planejamento mental para realizar uma determinada tarefa, considerando que deveremos executar um conjunto de passos até atingir o objetivo desejado.

Alguns exemplos de algoritmos do dia a dia são:

- Receitas culinárias;
- Manuais de instrução;
- Roteiros realização de tarefas específicas.

Observe a receita de bolo de fubá abaixo, que nada mais é do que um algoritmo preparado para a leitura e execução por um ser humano.

Separe os ingredientes:

- 2 colheres (sopa) de manteiga
- 2 xícaras (chá) de açúcar
- 4 ovos
- 1 xícara (chá) de leite
- 2 xícaras (chá) de fubá
- 1/2 xícara (chá) de farinha de trigo
- 1 colher (sopa) de fermento em pó
- Margarina e farinha de trigo para untar

Modo de preparo:

Na batedeira, bata a manteiga, o açúcar, os ovos e misture bem, até que fique um creme esbranquiçado. Em seguida, acrescente o leite, o fubá, a farinha e o fermento, misturando bem. Coloque em uma forma de buraco de 30 cm de diâmetro untada e enfarinhada. Leve ao forno médio, preaquecido, por 30 minutos, ou até que ao enfiar um palito, ele saia limpo. Retire do forno. Por fim, desenforme o bolo de fubá fofinho e sirva.

Você acha que um robô conseguiria seguir o passo a passo acima para fazer o bolo? Para que um computador possa desempenhar uma tarefa é necessário que esta seja detalhada, passo a passo, em uma linguagem compreensível pela máquina, por meio de um **programa (um algoritmo escrito em um formato compreensível pelo computador)**.

Quais são as propriedades essenciais de um algoritmo? Um algoritmo deve ser:

Completo	Todas as ações precisam ser descritas e devem ser únicas.
Sem redundância	Um conjunto de instruções só pode ter uma única forma de ser interpretada.
Determinístico	Se as instruções forem executadas, o resultado esperado será sempre atingido.
Finito	As instruções precisam terminar após um número limitado de passos.

Analise novamente a receita de bolo que colocamos acima. Ela contém todas as propriedades essenciais para um algoritmo computacional?

Formas de Representação

A forma de representação do algoritmo também é conhecida simplesmente por 'tipo de algoritmo'. É a maneira pela qual vamos apresentar o algoritmo. Existem três mais conhecidas:

Tipos de Algoritmo
Descrição Narrativa
Fluxograma
Pseudocódigo (Linguagem estruturada ou Portugol)

É bom saber que cada uma das formas de representação possui vantagens e desvantagens. Dessa forma, é de responsabilidade do programador escolher qual forma oferece as melhores características de acordo com a situação. É comum a combinação das representações, principalmente quando há a necessidade do entendimento por vários tipos de pessoas. Vamos analisar com atenção cada uma delas.

Descrição Narrativa

Os algoritmos são expressos diretamente em linguagem natural, a linguagem que usamos no nosso dia a dia, no nosso caso, o português. É parecido com o que vimos em relação à receita de bolo anteriormente. Vamos a um exemplo.

Você já usou um telefone público? Vamos imaginar como seria um algoritmo com descrição narrativa para utilizar um telefone público.

```
Início
  Tirar o fone do gancho;
  Ouvir o sinal de linha;
  Introduzir o cartão;
  Teclar o número desejado;
  Se der o sinal de chamar
    Conversar;
    Desligar;
    Retirar o cartão;
  Senão
    Desligar;
    Repetir;
  Fim Se.
Fim.
```

Quais os prós e contras de utilizar a descrição narrativa? Vejamos.

Prós	Contras
Não é necessário aprender novos conceitos, pois a língua natural já é bem conhecida.	A língua natural dá oportunidade para várias interpretações e ambiguidades, dificultando a transcrição desse algoritmo para um programa.

Fluxograma

Há muitos anos, o fluxograma tem aparecido como uma ferramenta interessante de representação do comportamento de programas, permitindo expressar:

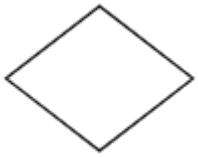
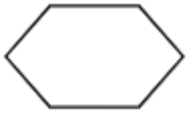
- o fluxo lógico da execução
- as operações envolvidas no processamento dos dados
- as entradas e saídas

Os fluxogramas são construídos a partir do uso de símbolos padronizados que expressam classes de operações comumente utilizadas nos programas. É mais preciso que a Descrição Narrativa, porém não se preocupa com detalhes de implementação do programa, tais como os tipos de variáveis utilizadas.

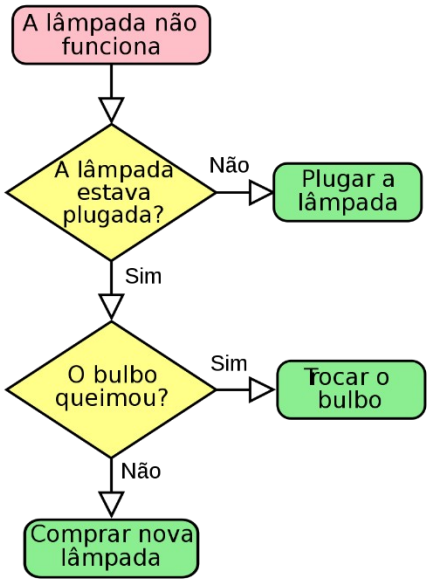
Quais os prós e contras de utilizar o fluxograma? Vejamos.

Prós	Contras
O entendimento de elementos gráficos é mais simples que o entendimento de textos.	Os fluxogramas devem ser entendidos e o algoritmo resultante não é detalhado, dificultando sua transcrição para um programa.

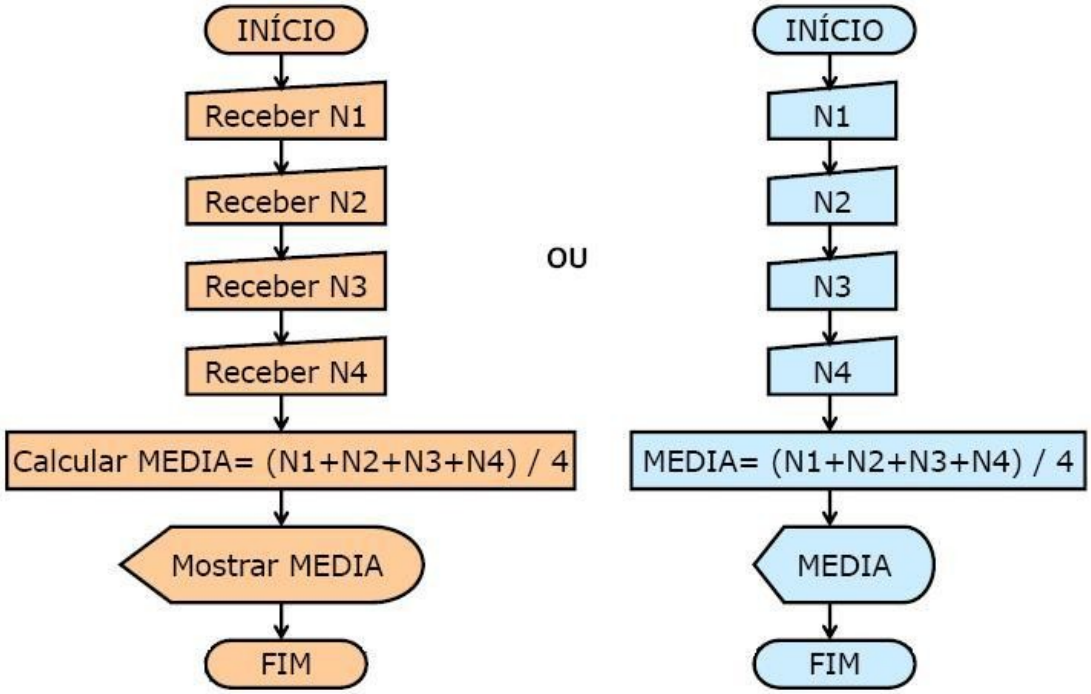
Seguem alguns símbolos utilizados quando criamos fluxogramas. Esses símbolos podem sofrer ligeira alteração de acordo com o autor e/ou com a ferramenta que está sendo utilizada.

Símbolo	Descrição
	**Terminal:** Representa o início e fim do fluxograma.
	**Processamento:** Representa a execução de operações ou ações como cálculos, atribuições de valores das variáveis, dentre outras.
	**Entrada de Dados:** Representa a entrada de dados para as variáveis através do teclado.
	**Saída de vídeo:** Através deste símbolo podemos representar a saída de informações (dados ou mensagens) por meio de um dispositivo visual de saída de dados, o monitor de vídeo e outros.
	**Decisão:** Representa uma ação lógica, que realizará uma sequência de instruções dependendo de uma condição verdadeira ou falsa.
	**Preparação:** Representa uma ação de preparação para o processamento, um processamento predefinido.
	**Conector:** Este símbolo é utilizado para interligar partes do fluxograma ou desviar o fluxo para um determinado trecho do fluxograma.
	**Conector de Página:** Utilizado para ligar partes do fluxograma em páginas distintas.
	Seta: Orienta a sequência de execução ou leitura, que poderá ser horizontal ou vertical.

Segue o Fluxograma representando o algoritmo de uma lâmpada que não funciona.



Como seria o fluxograma representando um algoritmo para calcular a média de quatro notas fornecidas? Vejamos.



Pseudocódigo

Os algoritmos são descritos em uma linguagem chamada pseudocódigo. Este nome é uma alusão à posterior implementação em uma linguagem de programação, ou seja, quando formos programar em uma linguagem, por exemplo Java, estaremos gerando código em Java. Por isso os algoritmos são independentes das linguagens de programação.

Ao contrário de uma linguagem de programação, não existe um formalismo rígido de como deve ser escrito o algoritmo.

O algoritmo deve ser fácil de interpretar e fácil de codificar, ou seja, ele deve ser o intermediário entre a linguagem falada e a linguagem de programação.

Estrutura Básica do Pseudocódigo

```
Algoritmo <nome_do_algoritmo>
  <declaração_de_variáveis>
Início
  <corpo do algoritmo>
Fim
```

Algoritmo	Palavra que indica o início da definição de um algoritmo em forma de pseudocódigo.
<nome_do_algoritmo>	Nome simbólico dado ao algoritmo com a finalidade de distingui-lo dos demais.
<declaração_de_variáveis>	Parte opcional onde são declaradas as variáveis globais usadas no algoritmo.
Início e Fim	Palavras que delimitam o início e o término, respectivamente, do conjunto de instruções do corpo do algoritmo.

Vamos ver alguns exemplos. Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida.

```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
```

```
   ESCREVA (NOME, MEDIA);
FIM.
```

Segue um Algoritmo que lê o raio de uma circunferência e calcula sua área.

```
ALGORITMO AREA_CIRCUNFERENCIA;
CONST
   PI = 3.1416;
VAR
   RAIO, AREA: REAL;
INICIO
   LEIA (RAIO); // PROCESSAMENTO
   AREA := PI * SQR(RAIO); // ENTRADA
   ESCREVA "AREA=" + AREA); // SAÍDA
FIM.
```

Quais os prós e contras de utilizar pseudocódigo? Vejamos.

Prós	Contras
Representação clara sem as especificações de linguagem de programação.	As regras do pseudocódigo devem ser aprendidas.

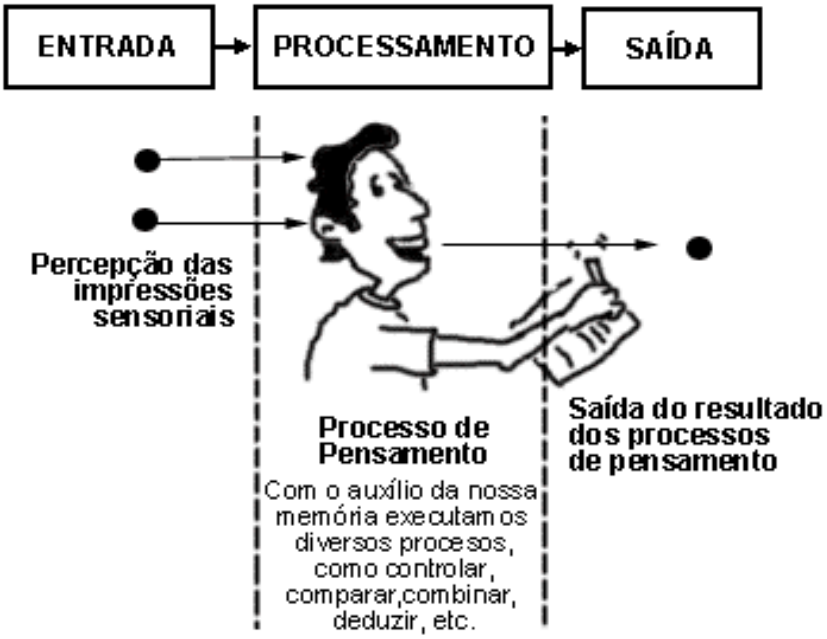
Fases do Algoritmo

Sabemos que o algoritmo é uma sequência lógica de instruções que podem ser executadas. Para facilitar o processo de criação do algoritmo, vamos dividi-lo em três fases.

- Entrada: são os dados necessários para a resolução do problema proposto.
- Processamento: são os procedimentos utilizados para chegar ao resultado.
- Saída: são os dados processados apresentando o resultado para o problema proposto.

Alguns autores utilizam a analogia com o próprio homem e vamos utilizar a mesma

analogia aqui com essa imagem que encontramos na Internet.



Vamos agora realizar essa divisão para observar se, de fato, ela traz uma maior facilidade para a criação do algoritmo. Vamos usar o mesmo algoritmo visto anteriormente para calcular a média final de um aluno.

```
ALGORITMO MEDIA_FINAL;
VAR
   NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;
   NOME: CARACTERE;
INICIO
   LEIA (NOME);
   LEIA (NOTA1, NOTA2, NOTA3, NOTA4);
   MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
   ESCREVA (NOME, MEDIA);
FIM.
```

Para montar o algoritmo proposto, faremos três perguntas:

1. Quais são os dados de entrada?

Resposta: os dados de entrada são NOME, NOTA1, NOTA2, NOTA3 e NOTA4

2. Qual será o processamento a ser utilizado?

Resposta: o procedimento será somar todos os dados de entrada numéricos e dividi-los por quatro.

3. Quais serão os dados de saída?

Resposta: o dado de saída será a média final.